Types of Programming Language

There are different types of **programming language**. We use the basic division of:

- high-level programming language
- low-level programming language

Most computer programs are written in high-level languages.

This stuff seems easy, but the answers needed are really precise.

What are high-level programming languages?

High-level programming languages include Python, C#, Visual Basic, Java and most of the other programming languages you might be familiar with.

Here's an example:

```
pWord = input("Enter the password").upper()
if pWord == "letmein":
   print("That's a silly password")
```

The special command words used in high-level languages are the **syntax**. That's why if you type something wrong you get a **syntax error**.

High-level languages look similar to written English. They can be understood quickly by anyone with some experience with the program – and the basic ideas in code might be able to be understood by a non-programmer.

What are low-level programming languages?

Low-level programming languages are ones which are much harder for human beings to understand – but are quicker for computers to execute. They include:

- Machine Code
- Assembly Language

The difference between machine code and assembly language is important to know.

Machine Code is the actual binary code executed by a computer's CPU. It looks like this:

```
00001010 00000 00000
10001101 00011 01000
00000010 00001 00010
```

Every program that gets written in any programing language needs to end up as machine code anyway – it's the only thing a CPU can process. It is possible for a human to program directly in machine code by writing out 1s and 0s, but this isn't very practical.

Because machine code is written in binary it's really hard for humans to write directly in. It can be done, but is far too difficult to write any sort of complex program in or to debug a program written in machine code.

Each make of CPU has its own set of machine code instructions. This means that a program written in machine code for one type of CPU will not work on machine using another CPU.

Programs are converted into machine code using **translators**. There are different types of translator – but that's a topic for another chapter.

Assembly language is a simple set of basic instructions that can be used to program. It looks like this:

```
MOVE R2, 0
ADD R2, R0, R2
SUB R1, R1, 1
HALT
```

You don't need to know what the commands mean, but you do need to be able to recognise assembly language.

Each assembly code instruction is very simple – in the example, numbers are added or subtracted. The R0, R1 or R2 is the register the data is stored in.

Each line of assembly code becomes one line of machine code – we say there is a **1:1 equivalence**. This makes it quick and easy to change assembly code into machine code, which means programs can be very quickly converted - so they run quicker.

This is a really important point to remember.

Assembly code is often used to program **embedded systems** which do highly specialised jobs. It is also used to control hardware components.

Embedded systems are computers in specialised devices such as washing machines.

Small programs can be written reasonably easily by humans because the key commands are similar to written English. But programs would be very long because each line can only do a very simple job.

Why use high-level programming languages

Because they are closer to written English, high-level languages are much easier for humans to understand. This means it ends up being easier and quicker to program and debug programs written in them.

This is a common exam question topic.

Advantages of using high-level languages:

- easier to understand, so quicker and easier to program and **debug**
- easier for other programmers to understand – so code can be **maintained** more easily
- access to **built in functions** (such as `print()`) – meaning a complex set of low-level processes can be turned into one command
- programs usually require **fewer lines of code** compared to assembly language
- access to **built in libraries** (such as random or time)
- easier to **structure** programs using iteration and selection
- can create **subroutines** to decompose code
- programs can be used on machines with different CPUs – so are **more portable** – they can be moved between makes of computer much more easily
- help and support more available – better documented, help libraries and more popular

Activities

a) Name the two types of programming language. Give examples of each.
b) How is machine code written? Why is it important?
c) Explain the relationship between assembly language and machine code.
d) Explain why a developer who is good at both types of program language would usually use a high-level language when writing a program.
e) Explain the circumstances in which they might use a low-level language.